支持差分隐私保护及离群点消除的并行 K-means 算法 *

樊一康，刘建伟†

(北京航空航天大学 电子信息工程学院，北京 100191)

摘 要：针对大数据环境下聚类分析的隐私保护问题，基于 MapReduce 计算框架，提出了一种并行化的支持差分隐私保护和离群点消除的 K-means 算法。算法并行地计算数据集中各点间的欧氏距离矩阵与最近邻超球半径以导出离群点的判定阈值，并在此基础上完成差分隐私保护下的初始聚类中心选取和并行聚类过程。理论分析证明整个算法满足 ε-差分隐私保护，实验结果说明该算法在隐私保护的有效性，聚类结果的可用性以及执行效率等方面取得了很好的平衡，相比于同类算法有较优的表现。

关键词：K-均值聚类；离群点消除；差分隐私；MapReduce

中图分类号：TP309.2 doi: 10.3969/j.issn.1001-3695.2017.12.0825

Parallel K-means algorithm with differential privacy preservation and outlier pruning

Fan Yikang, Liu Jianwei†

(*School of Electronic Information Engineering*, *Beihang University*, *Beijing* 100191, *China*)

Abstract: Aiming at the problem of privacy protection of clustering analysis in big data environment, based on the MapReduce computing framework, this paper proposed a parallel k-means algorithm that supports differential privacy protection and outlier elimination. The algorithm parallelly calculates the Euclidean distance matrix and nearest neighbor hypersphere radius between points in data set to derive the decision threshold of outliers, and then completes the initial cluster center selection and parallel clustering process under differential privacy protection. The theoretical analysis proves that the proposed algorithm satisfies ε-differential privacy, and the experimental results show that, compared with other algorithms, our algorithm performs better and has a good balance between the validity of privacy protection, the availability of clustering result and the efficiency of implementation.

Key words: K-means clustering; outlier pruning; differential privacy; MapReduce

0 引言

随着数据挖掘在各行各业越来越广泛的应用，挖掘引起的个人隐私泄露的问题也越来越受到重视。如何在保护个人隐私的同时为挖掘保持数据的可用性逐渐成为数据挖掘和信息安全领域的一个重要的研究方向。

早期典型的数据隐私保护模型是基于等价类（equivalence group）的 *k*-anonymity[1]及其一系列扩展模型，它们的核心思想都是将一个记录隐藏在一组记录之中[2]，保证任一记录与至少 *k*-1 个记录是不可区分的。但是这些模型总是需要针对新出现的攻击类型而作出相应的完善，例如针对“一致性”攻击而相继被提出的 *l*-diversity[3]、(*α*, *k*)-anonymity[4]、M-invariance[5]，针对“最小性”攻击而被提出的 m-confidentiality[6]。导致这一局面的根本原因是基于等价类的隐私保护依赖于对攻击者所掌握的背景知识的限定，而只要攻击者从限定之外的背景知识出发来攻击，模型往往就无能为力。除了需要被动地针对新型攻击而发展变种的缺陷外，这些模型也未能提供一个严格的量化方法来度量其隐私保护水平，这也影响了它们的鲁棒性。

在这样的背景下，Dwork 提出了差分隐私（differential privacy，DP）[7]的概念。差分隐私保护通过添加噪声的方式可使得处理数据集的某个算法的输出对数据集中某条记录的变化甚至存在与否不敏感。因此，个人的一条记录被包含在某个提供差分隐私保护的数据集中而带来的隐私泄露风险可以被控制在一个可接受的范围内，即便是在攻击者拥有最大背景知识(即攻击者已知除目标记录外的所有记录的信息)的情况下。同时，差分隐私定义了严格的攻击者模型并提供了可证明的量化的评估方法来度量隐私泄露风险。而基于差分隐私保护的算法在输出上需要添加的噪声独立于数据集的规模，且少量的噪声即可达到较高的隐私保护水平并保持挖掘结果的可用性。正因为差分隐私在上述几个方面有着良好的表现，其迅速被业界接纳并

收稿日期：2017-12-27；**修回日期：**2018-02-27 **基金项目：**国家自然科学基金资助项目（61272501）

作者简介：樊一康（1992-），男，硕士，主要研究方向为信息安全、数据挖掘；刘建伟（1964-），男（通信作者），教授，博士，主要研究方向为信息安全、网络安全、密码学（liujianwei@buaa.edu.cn）.

在近年来成为隐私保护领域的研究热点。

目前基于差分隐私的数据挖掘研究工作在关联规则挖掘上的积累较多，而在聚类分析上的积累则相对较少。K-means 算法是公认的应用最广泛的聚类算法之一，对于如何在其聚类分析的准确性与隐私保护的有效性间取得平衡，以及对其执行效率的优化上，国内外学者都做了一些积累与贡献。文献[8]给出了一种提供差分隐私保护的 K-means 算法，文献[9]从隐私保护预算分配的角度对文献[8]中的工作进行了完善，文献[10]基于 MapReduce 计算框架[11]将文献[9]完善的算法并行化，但它们均没有考虑算法面对噪声及离群点的鲁棒性。文献[12]提出了 OEDP K-means 算法，其考虑了离群点对 K-means 算法的消极影响并引入了一种检测离群点并消除的方法，但该方法的时间复杂度为 $O(N^2)$，N 为数据集中的记录数，其对算法执行效率的影响在对大数据集进行聚类分析时尤为明显，此外，OEDP 算法对初始聚类中心的选取策略计算复杂高且在某些数据分布的情况下会发生劣化而导致更慢的收敛速度。文献[13]提出了另一种基于差分隐私保护的 IDP K-means 算法，其考虑了初始聚类中心的选择对算法的影响，但其对初始聚类中心的选择仅是简单的分段取平均值，并没有排除可能的离群点带来的消极影响。另外，OEDP k-means 算法和 IDP k-means 算法都是串行结构，而并行化的算法结构对大规模数据集上的聚类需求是很有意义的。

基于此，本文提出了一种支持差分隐私保护及离群点消除的并行 K-means 聚类方法（因其关注 Parallelization 与 Pruning 以及 Privacy，可简称为 TripleP K-means 聚类方法），其基于 MapReduce 计算框架并行地计算数据集中各点间的欧氏距离矩阵与最近邻超球半径以导出离群点的判定阈值，并在此基础上完成差分隐私保护下的初始聚类中心选取和并行聚类过程。本文给出了算法的差分隐私性证明与实验分析，相比于同类算法，本文算法在隐私保护的有效性，聚类结果的可用性以及聚类效率和扩展性等方面取得了较好的平衡。

1　相关概念

1.1　差分隐私的定义及性质

差分隐私[7,14]模型提供了严格且可证明的隐私保护定义以及对泄露风险的量化评估方法。因其对隐私的保护不依赖于对攻击者所拥有的背景知识的限定，从而可应对任意背景知识下的攻击行为。其主要方式是在数据集中或者在算法输出上添加随机噪声来实现隐私保护，同时保持原有数据及所含信息的可用性。具体地，差分隐私模型的一些定义和性质如下所述：

1.1.1 差分隐私

设有两个数据集 D 和 D'，二者的属性结构相同，记二者的对称差为 $D\Delta D'$，记对称差中的记录数量为 $|D\Delta D'|$。称 $|D\Delta D'|=1$ 时的 D 和 D' 为邻近数据集（Adjacent Dataset）。

设 $\mathcal{K}$ 为邻近数据集 D 和 D' 上的一个随机算法，记 $\mathcal{K}$ 所有可能的输出的集合为 $Range(\mathcal{K})$，记事件 X 发生的概率为 $P_r[X]$，则对 $Range(\mathcal{K})$ 的任一子集 S，若 $\mathcal{K}$ 满足

$$P_r[\mathcal{K}(D)\in S]\le \exp(\varepsilon)\times P_r[\mathcal{K}(D')\in S] \tag{1}$$

则称算法 $\mathcal{K}$ 满足 ε- 差分隐私保护。称 ε 为隐私保护预算，它限制了数据集中某一记录的变化对算法输出的影响，越小的 ε 意味着越严格的隐私保护。一般 ε 的取值范围是 $(0.01,0.1)$ 或在某些情况下为 $(\ln 2,\ln 3)$ [14]。

L_1 敏感度

设函数 $f:D\to R^d$，其将数据集 D 映射到 d- 维实数空间中的一个向量，对于任意的邻近数据集 D 和 D'，令

$$\Delta f=\max_{D,D'}\|f(D)-f(D')\|_1 \tag{2}$$

则称 Δf 为函数 f 的 L_1 敏感度[7,14]，其中 $\|f(D)-f(D')\|_1$ 是 $f(D)$ 与 $f(D')$ 间的 1- 阶范数距离。注意 L_1 敏感度独立于数据集而仅由函数本身决定，它是影响所需加入的随机噪声量的关键参数。

Laplace 机制

差分隐私保护的噪声实现机制主要有指数机制和 *Laplace* 机制[15]两种，相比于指数机制，*Laplace* 机制对数值型数据的保护更为适合，因此多用于构造支持差分隐私保护的聚类算法。记位置参数为 0、尺度参数为 b 的 *Laplace* 分布为 $Lap(b)$，则其概率密度函数为

$$p(x\,|\,b)=\frac{1}{2b}\exp(-\frac{|x|}{b}) \tag{3}$$

对于函数 $f:D\to R^d$，其 L_1 敏感度为 Δf，若随机算法 $\mathcal{K}$ 有

$$\mathcal{K}(D)=f(D)+(Lap_1(\Delta f/\varepsilon),Lap_2(\Delta f/\varepsilon),...,Lap_d(\Delta f/\varepsilon)) \tag{4}$$

则算法 $\mathcal{K}$ 满足 ε- 差分隐私保护，其中 $Lap_i(\Delta f/\varepsilon),i\in[1,d]$ 之间是相互独立的。由 $p(x\,|\,b)$ 及 $b=\Delta f/\varepsilon$ 可知，越小的 ε 需要引入的噪声就越大。

1.1.2 序列组合性

设有 n 个随机算法 $\mathcal{K}_1,\mathcal{K}_2,...,\mathcal{K}_n$，其隐私保护预算分别为 $\varepsilon_1,\varepsilon_2,...,\varepsilon_n$，则对于某一数据集 D，由这 n 个随机算法序列组合成的算法 $\mathcal{K}(\mathcal{K}_1(D),\mathcal{K}_2(D),...,\mathcal{K}_n(D))$ 满足 $(\sum_{i=1}^{n}\varepsilon_i)$- 差分隐私保护，此即差分隐私的序列组合性[16]。

1.1.3 并行组合性

设有 n 个随机算法 $\mathcal{K}_1,\mathcal{K}_2,...,\mathcal{K}_n$，其隐私保护预算分别为 $\varepsilon_1,\varepsilon_2,...,\varepsilon_n$，则对 n 个交集为空集的数据集 $D_1,D_2,...,D_n$，由着 n 个随机算法并行组合成的算法 $\mathcal{K}(\mathcal{K}_1(D_1),\mathcal{K}_2(D_2),...,\mathcal{K}_n(D_n))$ 满足 $(\max_i \varepsilon_i)$- 差分隐私保护，此即差分隐私的并行组合性[16]。

1.2　k-means 聚类算法与离群点

对一个 $\mathbb{R}^T$ 上的数据集 $\mathcal{X}=\{x_1,x_2,...,x_N\}$，其包含 N 个具有 T 维属性值的数据点，聚类算法可将其划分为 K 个簇：$\mathcal{P}=\{P_1,P_2,...,P_K\}$ 并满足 $\bigcup_{i=1}^{K}P_i=\mathcal{X},P_i\cap P_j=\phi(1\le i\ne j\le K)$，簇内

的点或称向量之间高度相似而簇间的相异。k-means 算法被公认是应用最广泛的聚类算法之一，其首先选择 K 个初始中心点，然后将数据集中的点按平方欧氏距离分配到最近的中心点从而形成 K 个簇并调整各中心点到各簇的平均值，算法重复这一分配调整的过程直至满足收敛判据或者迭代次数达到上限。k-means 算法的一个重要优势是时间及空间复杂度均与 N,T,K 呈线性相关（一般地，$T \ll N, K \ll N$）。但因为 k-means 算法以平方欧氏距离度量相似度，所以其对数据噪声及离群点十分敏感，少数的离群点就能对其所属簇的中心造成很大的影响，在降低聚类准确性的同时也可能致使 k-means 算法迭代次数增加而降低算法的运行效率。为回避离群点带来的消极影响，可以选择鲁棒性更好的距离函数如 City-block ($\mathcal{L}_1$) [17]，但更直接的处理方法是离群点修剪。有关离群点的一些定义及概念如下述。

1.2.1 r-最近邻超球及其半径

对于 $\mathbb{R}^T$ 上的数据集 $\mathcal{X}=\{x_1,x_2,...,x_N\}$，称任一数据点 x_i 与离其最近的 r 个数据点在 T 维空间中构成的超球为 x_i 的 r-最近邻超球，这一概念的原型为文献[12]中的 r-最近邻区域（r-nearest-neighbour area）。

希望点 x_i 的 r-最近邻超球的大小可以反映出点 x_i 及其周围点的稀疏程度，因此取点 x_i 与其超球中的所有点的平均欧氏距离为超球的半径，记数据点 x_i 与其 r-最近邻超球中的点 $n_l(1 \le l \le r)$ 间的欧氏距离为 $dist(x_i,n_l)$，超球半径为

$$rRadius(x_i,\mathcal{X}) = [\sum_{l=1}^{r} dist(x_i,n_l)]/r \quad (5)$$

1.2.2 离群点及判定阈值

离群点[12]是指与其他点有明显不同的数据点，一般可分为全局离群点（global outlier）、情境离群点（contextual outlier）、集体离群点（collective outlier），鉴于本文主要面对的是数值型数据的分析处理，不涉及具体情境或者群组，因此只考虑全局离群点，也即数据集中与其余点差别均很大的数据点。对于离群点的判定，由数据集中的稀有类由至多 5%的数据点组成[18]，估计数据集 $\mathcal{X}$ 中离群点数目不超过 $N \times 0.05$，记为 O_{num}，记离群点的 r-最近邻超球半径的判定阈值为 $\mathcal{T}$，取其值为数据集中 r-最近邻超球半径最大的 O_{num} 个点的半径均值，从而认为 r-最近邻超球半径超过阈值 $\mathcal{T}$ 的数据点为离群点。

2　算法设计与分析

本文提出的算法基于 MapReduce 计算框架，并行地计算得到离群点的判定阈值，并在此基础上完成差分隐私保护下的初始聚类中心选取和并行聚类分析。其中离群点相关的处理可以提高 K-means 算法的准确度和运行效率，而差分隐私保护则关注在数据集中的一条记录改变时，算法聚类结果（各聚类质心位置及所含数据项的数目）的相应变化不会泄露隐私信息。具体来说，算法分为三个阶段：

首先是准备阶段。对于一个 $\mathbb{R}^T$ 上的数据集 $\mathcal{X}=\{x_1,x_2,...,x_N\}$，由前文离群点的相关概念可知，离群点的相关计算依赖于 $\mathcal{X}$ 中各数据点间的欧氏距离 $dist(x_i,x_j)(1 \le i \ne j \le N)$，用一个 $N \times N$ 的距离矩阵 $Dist$ 来表示它。而在计算该距离矩阵前，需要对数据集进行归一化处理。因此首先需要计算数据集中 T 个属性各自的最大值和最小值。然后完成数据的归一化以及按一定方式分拆与整理数据集中的所有数据以适配 MapReduce 下对各数据点间欧氏距离的并行计算，最终获得距离矩阵 $Dist$。

其次是过渡阶段，算法完成离群点判定阈值的计算以及初始聚类中心的选取。由上一阶段得到的距离矩阵 $Dist$，并行地计算数据集中各数据点的 r-最近邻超球半径 $rRadius(x_i,\mathcal{X})$，进而计算出离群点的判定阈值 $\mathcal{T}$。此外，考虑到 K-means 算法因其梯度下降的性质而对初始中心点的选择十分敏感[17]，盲目选择初始中心可能导致第三阶段的聚类算法迭代次数增多而影响效率，且过慢的收敛还会引入更多的噪声而影响聚类结果的可用性。因此算法利用距离矩阵 $Dist$ 中的信息完成初始聚类中心的选取，核心策略是让这 K 个初始聚类中心能尽可能地分别落在假设存在的 K 个簇中，并在选取过程中剔除离群点。

算法在第三阶段完成数据的聚类分析及其中的差分隐私保护。由第二阶段得到的 K 个初始聚类中心出发，将数据集 $\mathcal{X}$ 中的数据点参考其与各聚类中心的距离大小就近划分成 K 个簇，并计算各簇中数据点的数量及总和并添加相应量的 $Laplace$ 噪声，进而获得新一代的 K 个聚类中心。算法重复这一过程，直至满足收敛判据或者迭代次数达到上限。最终算法输出最后一次迭代得到的聚类中心及所含数据点的数量。

2.1　具体步骤

2.1.1 计算最大值与最小值

设 $\mathbb{R}^T$ 上的数据集 $\mathcal{X}=\{x_1,x_2,...,x_N\}$ 被均匀划分为 M 个数据片，每个数据片中最多有 $\lceil N/M \rceil$ 个数据点，记第 m 个数据片及其索引分别为 $Split_m$ 和 $SplitIndex_m$。记由 T 个属性域的理论最大值和理论最小值构成的数据点分别为 $Attr_{MAX}$ 和 $Attr_{MIN}$，记由数据集 $\mathcal{X}$ 在 T 个属性域上的最大值和最小值构成的数据点分别为 x_{MAX} 和 x_{MIN}，记由数据片 $Split_m$ 在 T 个属性域上的最大值和最小值构成的数据点分别为 $x_{m,\max}$ 和 $x_{m,\min}$。

Master（主节点）指派 M 个 Map 任务与 2 个 Reduce 任务。需要注意的是，在经典形式中，Map 任务会遍历数据片中的各键值对并在其上分别调用 Map 函数，而在这里采取相异的策略以便于处理数据，即将上述遍历过程放在 Map 函数之中（通过重载 mapper 类的用户继承类中的 run 方法来实现），因此每个 Map 任务只需调用一次 Map 函数，在后续的一些步骤中也用到了同样的策略，不再一一说明。本文参考 MapReduce 框架提出者在文献[11]中的表达形式，在 Map 函数中用 EmitIntermediate() 函数表示输出中间键值对（这些中间键值对会由框架传递给 Reduce 任务），在 Reduce 函数中用 Emit()函数表示输出结果键值对（写入到每个 Reduce 任务的输出文件中），在后续的各步骤中依然保持这样的表达形式。

具体地，计算最大值与最小值的 Map 函数及 Reduce 函数的伪代码如下所示:

```
map(Index SplitIndex_m, DataSplit Split_m):
    x_{m,max} = Attr_MIN;
    x_{m,min} = Attr_MAX;
for each x_i in Split_m:
update x_{m,max} and x_{m,min};
EmitIntermediate("max", x_{m,max});
EmitIntermediate("min", x_{m,min});
reduce(String type, Iterator vectors)
//type 取值为"max"或"min"
//vectors 是同 type 下各 x_{m,max} 或 x_{m,min} 的集合
if (type.equals("max"))
for each x_{m,max} in vectors:
update x_MAX;
Emit("max", x_MAX);
else
for each x_{m,min} in vectors:
update x_MIN;
Emit("min", x_MIN);
```

2.1.2 归一化及倒排索引

延用上一步中划分得到的 M 个数据片。记数据集 $\mathcal{X}$ 的 T 个属性域的集合为 $Attributes=\{A_1,A_2,...,A_T\}$，记数据点 x_i 在属性域 $A_t(1\le t\le T)$ 上的分量为 a_{t,x_i}，记其归一化值为 a'_{t,x_i}，将数据片 $Split_m$ 中的点 x_i 及其在属性域 A_t 上的归一化分量作为键值对 $<x_i,a'_{t,x_i}>$ 存放在一个关联数组中，记为 $H_{t,m}$，同理记数据集 $\mathcal{X}$ 中所有点在属性域 A_t 上的相应关联数组为 H_t。Master 指派 M 个 Map 任务与 T 个 Reduce 任务。用 Map 函数将 $Split_m$ 中的数据在归一化的同时按各属性域拆分，用 Reduce 函数将属性域下各数据片的数据整合，二者伪代码如下所示：

```
map(Index SplitIndex_m, DataSplit Split_m):
for each A_t in Attributes:
H_{t,m} =new ssociativeArray();
//新建一个关联数组 H_{t,m}
for each x_i in Split_m:
    H_{t,m}[x_i] = (a_{t,x_i} - a_{t,x_MIN}) / (a_{t,x_MAX} - a_{t,x_MIN});
EmitIntermediate (A_t, H_{t,m});
reduce(String A_t, Iterator arrays):
// arrays 是各数据片在属性 A_t 下 H_{t,m} 的集合
    H_t =new ssociativeArray();
for each H_{t,m} in arrays:
Merge(H_t, H_{t,m});
Emit(A_t, H_t);
```

2.1.3 计算欧氏距离矩阵

如果 $T\le M$（此时往往有 $\lceil M/T\rceil\le N$），为了达到更优的扩展性，可以对上一步获得的各属性域下的关联矩阵进行复制及划分。具体地，记 $Q=\lceil M/T\rceil$，对于某个关联矩阵 $H_t=[(x_1,a'_{t,x_1}),(x_2,a'_{t,x_2}),...,(x_N,a'_{t,x_N})]$，将其中的键值对均匀划分为 Q 个部分，每部分至少包含 $\lfloor N/Q\rfloor$ 个键值对，从而由 H_t 派生出 Q 个内容与 H_t 一致但下标不同的关联矩阵 $\{H_{t,1},H_{t,2},...,H_{t,Q}\}$。而对于其中某个 $H_{t,q}(1\le q\le Q)$，用它的属性域标号 t 与键值对序号 q 的乘积 tq 作为它的索引，Map 函数只计算该关联矩阵中第 q 部分键值对涉及的点与其余点在属性域 A_t 下的距离分量。为了节省空间，对于 Map 函数的输出，不同于传统中间键值对（intermediate key/value pair）的形式 $<(x_i,x_j),value>$，采用"带"(stripe）的形式 $<x_i,[(x_j,value)...]>$ 来存放，即将点 x_i 与其余点在属性域 A_t 下的距离分量存放在一个关联数组 $D_{t,i}$ 中。以带的形式来组织数据相比于中间键值对可以节省约三分之一的空间，并且在利用 MapReduce 框架中的 Combiner 时也能有更高的合并效率。Reduce 函数将数据点 x_i 在各属性域下与其余点的距离分量整合并计算得到其与其他点的欧氏距离最后存入关联数组 D_i 之中。如果 T 与 M 较为接近，则取 Q 为 1，即不再对 H_t 分段划分而作为一整段来处理。Master 共指派 $T\times Q$（其值可能略大于 M）个 map 任务和 M 个 reduce 任务。到此，即可获得距离矩阵 $Dist$，其由 M 份 Reduce 任务输出的数据文件组成。Map 函数及 reduce 函数的伪代码如下所示：

```
map(Index tq, AssociativeArray H_{t,q}):
for each x_i in the q_th segment of H_{t,q}:
    D_{t,i} =new ssociativeArray();
for each x_j (j≠i) in H_{t,q}:
    D_{t,i}[x_j] = (a'_{t,x_i} - a'_{t,x_j})^2;
EmitIntermediate (x_i, D_{t,i});
reduce(Point x_i, Iterator arrays):
// arrays 是点 x_i 在各属性域下 D_{t,i} 的集合
    D_i =new ssociativeArray();
for each D_{t,i} in arrays:
AttributeSum(D_i, D_{t,i});
//将各属性域下的距离平方求和
AttributeSquareRoot(D_i);//开方求欧氏距离
Emit(x_i, D_i);
```

2.1.4 计算判定阈值

上一步得到的距离矩阵 $Dist$ 由 M 段数据文件组成，记第 m 段数据文件及其索引分别为 $Segment_m$ 和 $SegmentIndex_m$。对于某段数据文件 $Segment_m$，Map 函数线性扫描其中各数据点与其余数据点的距离以获得最小的 r 个欧式距离，并由此计算出各点的 r-最近邻超球半径 $rRadius(x_i,\mathcal{X})$，同时 Map 函数线性扫描该段数据文件中各数据点的最近邻超球半径以获得最大的 O_{num} 个值并作为一个集合 Set_m 输出。Reduce 函数线性扫描所有其收到的集合并选取其中最大的 O_{num} 个值，取其均值作为判定阈值 T。Master 指派 M 个 Map 任务和 1 个 Reduce 任务。Map

函数及 Reduce 函数的伪代码如下所示:

```
map(Index SegmentIndex_m, DistSplit Segment_m):
Set_m=new Set();
//新建含 O_num 个 0 的集合
for each x_i in Segment_m:
scan D_i to get the r smallest distance;
calculate rRadius(x_i, X);
if (it >minimum( Set_m))
replace the minimum with it;
EmitIntermediate( X, Set_m);
reduce(Dataset X, Iterator sets)
// sets 是各段数据的 Set_m 的集合
Set = new Set();
//新建含 O_num 个 0 的集合
for each Set_m in sets:
for each r in Set_m:
if (r >minimum(Set))
replace the minimum with r;
T=mean(Set);//求集合中半径的均值
Emit( X, T);
```

2.1.5 选取初始聚类中心

经过上一步不仅获得了离群点的判定阈值 $\mathcal{T}$，也得到了各数据点的 r- 最近邻超球半径 $rRadius(x_i,\mathcal{X})$。记 K 个初始簇及聚类中心分别为 $\{C_1,C_2,...,C_K\}$，$\{\mu_1,\mu_2,...,\mu_K\}$，算法从数据集 $\mathcal{X}$ 的 N 个数据点中随机选取一个非离群点 μ_1' $(rRadius(\mu_1',\mathcal{X})<\mathcal{T})$，并根据距离矩阵 $Dist$ 中的信息从其余数据点中拿出距离点 μ_1' 最近的 $\lfloor N/K \rfloor$ 个非离群点（遇到离群点则剔除）与点 μ_1' 组成初始簇 C_1，接着算法从剩余的数据点中再次随机选取一个非离群点作为 μ_2' 并同样拿出其周围的一部分点与其组成初始簇 C_2，算法重复这一拿取的过程直至获得 K 个初始簇。然后计算各簇中数据点的总和，并向总和与簇中数据点数目（约 $\lfloor N/K \rfloor$）分别添加 $Laplace$ 噪声，进而求各簇的均值点作为初始聚类中心。由 Master（主节点）整合距离矩阵 $Dist$ 并单独执行这一步算法。

2.1.6 聚类分析及隐私保护

经过上一步得到了 K 个初始聚类中心 $\{\mu_1,\mu_2,...,\mu_K\}$ 与不含离群点的数据集 $\mathcal{X}'$，仍划分其为 M 个数据片，第 m 个数据片及其索引仍记为 $Split_m$ 和 $SplitIndex_m$。对某个数据片 $Split_m$，Map 函数计算其中数据点与各聚类中心的距离并就近关联片中数据点到相应的聚类中心，并以 $<\mu_k,C_{k,m}>$ 的形式发布各中心点与片中数据点（其集合记为 $C_{k,m}$）的关联关系。Reduce 函数则整合与各中心点关联的所有数据点为一簇并计算其数目与总和，记第 k 个中心点的簇为 C_k，簇中数据点数目及总和分别为 num_k 和 sum_k，然后向 num_k 和 sum_k 中添加相应量的 $Laplace$ 噪声得到 num_k' 和 sum_k'，进而计算出簇 C_k 的均值点作为新一代的聚类中心 μ_k。Master 接收 Reduce 任务输出的新一代 $\{\mu_1,\mu_2,...,\mu_K\}$ 并结合其在上一轮的值计算是否满足收敛条件以及判断迭代次数是否达到上限，进而决定是进入下一轮迭代还是终止算法并给出聚类结果。Master 共指派 M 个 Map 任务和 K 个 Reduce 任务。Map 函数及 Reduce 函数的伪代码如下所示:

```
map(Index SplitIndex_m, DataSplit Split_m):
for k from 1 to K:
C_{k,m}=new Set();
//为中心点 μ_k 建立一个空的集合
for each x_i in Split_m:
find the nearest centre μ_k;
add x_i to μ_k's C_{k,m};
for k from 1 to K:
EmitIntermediate( μ_k, C_{k,m});
reduce(Point μ_k, Iterator sets)
// sets 是各段数据 C_{k,m} 的集合
for k from 1 to K:
C_k =new Set();
//为中心点 μ_k 建立一个空的集合
for each C_{k,m} in sets:
Merge( C_k, C_{k,m});
Calculate num_k and sum_k of C_k;
Add laplace noise and get num_k' sum_k';
update μ_k by sum_k' / num_k';
Emit( μ_k, num_k');
```

2.2 差分隐私分析

由上述，在聚类分析的每一次迭代过程中，数据集 $\mathcal{X}$ 的 M 个数据片上各有一个 Map 任务，Map 任务的输出又被划分为 K 个部分并行地由对应的 Reduce 任务完成数据的整合与新一代聚类中心的计算。因此每一次迭代得到的聚类结果是各 Reduce 任务输出的并行组合，设第 l 次迭代各 Reduce 任务的输出都可以满足 ε_l- 差分隐私保护，由前文所述的差分隐私的并行组合性，则第 l 次迭代的输出可以满足 ε_l- 差分隐私保护，而聚类分析的最终结果相当于多次迭代过程的序列组合的输出，设总迭代次数为 L，由差分隐私的序列组合性，算法的最终输出满足 $(\sum_{l=1}^{L}\varepsilon_l)$- 差分隐私保护。本文采用文献[9]的隐私保护预算分配策略，设总的预算为 ε，则给予每一次迭代剩余隐私预算

的一半，即取第 l 次迭代的隐私保护预算 ε_l 为 $\varepsilon/2^l$，如此不论算法做了多少次迭代，总有 $(\sum_{l=1}^{L}\varepsilon_l)<\varepsilon$。需要注意的是，应将初始簇的选取及初始聚类中心的生成看做第一次迭代。

由第 l 次迭代的隐私保护预算 ε_l，可以计算出聚类中心点 μ_k 的簇 C_k 中数据点数目 num_k 及总和 sum_k 需要添加的 $Laplace$ 噪声量的大小。首先分析 num 及 sum 的全局敏感度，考虑修改或者删去 $\mathbb{R}^T$ 上归一化的数据集 $\mathcal{X}=\{x_1,x_2,...,x_N\}$ 中的一点，易知 num 的全局敏感度 $\Delta f_{num}=1$，而因为归一化，$x_i\in[0,1]^T$，由此知 sum 的全局敏感度 $\Delta f_{sum}=T$，因此聚类结果（或称查询序列）的 $\Delta f=(T+1)$。所以第 l 次迭代 num_k 及 sum_k 需要添加的 $Laplace$ 噪声量应为 $Lap((T+1)\times 2^l/\varepsilon)$。

综上，本文算法可以达到 ε- 差分隐私保护。

3　实验分析

本文算法的隐私性已经在上文中得到证明，因此实验部分主要考察算法的运行效率及其聚类结果的可用性。实验环境由作为主节点的一台计算机和作为分节点的三台计算机组成。每台计算机的 CPU 为双核 3.3GHz，内存 4GB，操作系统为 Ubuntu 并部署了 Hadoop2.6.0，算法实现语言为 Java。实验用数据集为 UCI Knowledge Discovery Archive database 中的 Ecoli 数据集（记录数为 336，属性数为 8，中心数为 8），Wine 数据集（记录数为 178，属性数为 13，中心数为 3），HTRU2 数据集（记录数为 17898，属性数为 9，中心数为 2），这三个数据集都适用于分类与聚类算法以及离群点检测算法的检验。在检验算法之前首先对这些数据集进行一些合理的预处理如对记录点的去重和各点属性值的归一化（min-max 标准化）。实验中与本文算法作对比的是 OEDP k-means[12]，IDP k-means[13]，DP k-means[9] 等目前基于差分隐私保护的聚类算法。

3.1　聚类结果可用性的评估

考虑到实验所用的各数据集都已给出了参考分类，本文采用 F-$measure$ 值来评估各算法聚类结果的可用性。F-$measure$ 综合了信息检索与挖掘中的准确率(precision)和召回率(recall)，相比于其他一些评估方法更为中肯。假设某数据集中的记录总数为 N，标准聚类结果为 $\mathcal{P}=\{P_1,P_2,...,P_K\}$，算法的实际聚类结果为 $\mathcal{P}'=\{P_1',P_2',...,P_K'\}$，$|P_i|$ 与 $|P_i'|$ 分别为 P_i 和 P_i' 中的记录数目，$|P_i\cap P_i'|$ 为 P_i 和 P_i' 交叉的记录数目，记第 $i(1\le i\le K)$ 个聚类的准确率和召回率分别为 $Prec_i$ 和 Rec_i，则：

$$Prec_i=\frac{|P_i\cap P_i'|}{|P_i|},Rec_i=\frac{|P_i\cap P_i'|}{|P_i'|} \tag{6}$$

二者的加权调和平均（取准确率与召回率同样的权重）为

$$F_i=\frac{2\cdot Prec_i\cdot Rec_i}{Prec_i+Rec_i} \tag{7}$$

然后对各 F_i 取加权平均即为算法聚类结果的可用性度量

$$F\text{-}measure=\sum_{i=1}^{K}\frac{|P_i|}{N}F_i \tag{8}$$

F-$measure$ 的取值范围是 [0,1]，值越大则说明聚类结果的可用性越高。

3.2　算法参数选取

首先，对于隐私保护预算 ε，一般在 [0.05,1] 中取值，因此线性地在该区间中取值来观察其对算法的可用性和执行效率的影响。

其次，对于聚类数 K，因为实验中用到的各数据集都已给出了参考分类，而本文主要关注的是 k-means 算法中的离群点消除、初始中心点选取和对差分隐私的支持，因此取各数据集参考分类的数目为其聚类数 K。

最后，在本文算法和 OEDP k-means 算法对离群点的处理过程中，都涉及 r- 最近邻区域的概念并对参数 r 较为敏感。参考文献[19]中的方法以 F-$measure$ 为目标函数对参数 r 进行调整并取最优，即对 Ecoli 数据集取其 r 为 3，对 Wine 数据集取其 r 为 1，而对 HTRU2 数据集取其 r 为 7，并在后续的实验中将各数据集 r 值代入本文算法与 OEDP k-means 算法。

3.3　可用性与运行效率对比及分析

首先，在 Ecoli 数据集和 Wine 数据集上单机运行了本文算法，OEDP k-means 算法，IDP k-means 算法和 DP k-means 算法。对每一次在 [0.05,1] 中线性取得的隐私保护预算 ε，都重复执行上述四种算法 5 次，以取得各算法的 F-$measure$ 值和执行所耗时间的平均值。考虑到 IDP k-means 算法并不包含对离群点的处理而 DP k-means 算法不包含任何附加的初始化策略，这里的执行耗时是在完成聚类初始化（如可能的离群点消除与初始聚类中心的选取）后各算法聚类过程所耗的时间。具体情况如下图 1 到图 4 所示。

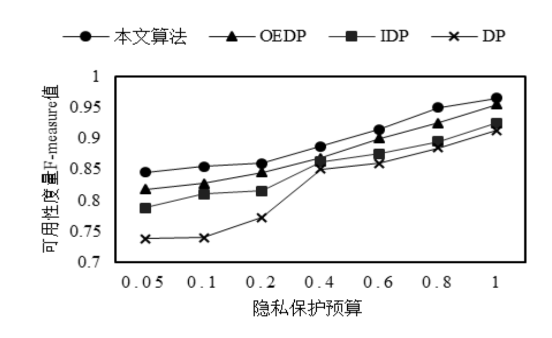


图 1　Ecoli 数据集上各算法的可用性

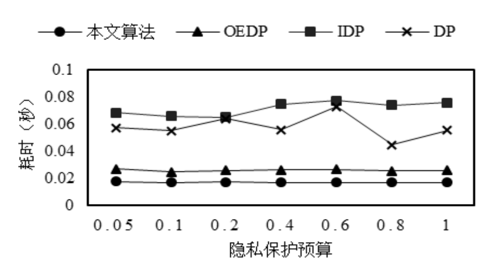


图 2　Ecoli 数据集上各算法的执行效率

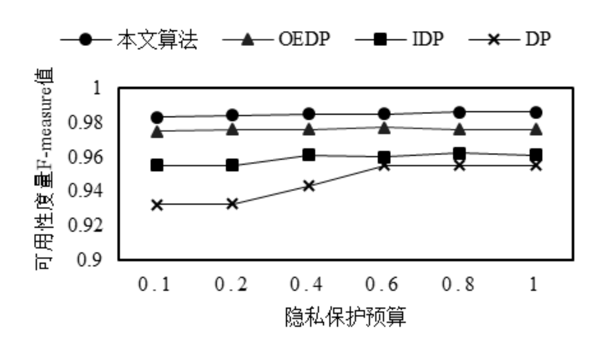


图 3　Wine 数据集上各算法的可用性

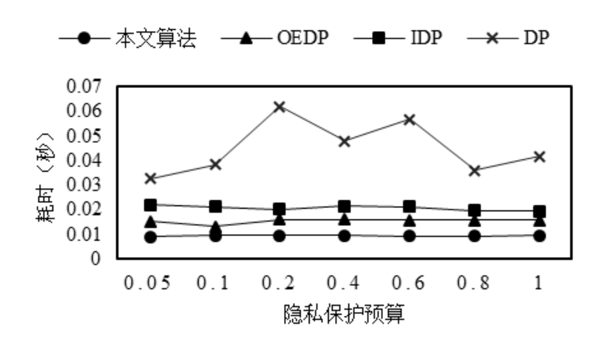


图 4　Wine 数据集上各算法的执行效率

由图 1 到图 4 可知，对隐私保护预算 ε 的每一个值，本文提出的算法相比于其他算法都有着更高的 F-measure 值和更少的聚类耗时；并且当 ε 变化时，本文算法耗时的波动相比其他算法较小；另外，F-measure 值代表的聚类结果可用性随着 ε 的增加隐私保护水平的降低而提高。

从算法设计的角度而言，相比于一般的 DP k-means 算法，本文算法在提供差分隐私保护的同时，对离群点和初始聚类中心的处理提升了聚类结果可用性；对于离群点的判定，本文主要利用的是数据点的最近邻超球半径，而 OEDP k-means 算法所用的最近邻密度实际上是最近邻超球半径的倒数，二者在导出判定阈值上没有本质区别，因此省去倒数计算的本文算法更为简洁，此外本文算法导出判定阈值的过程是并行化的；对于初始聚类中心的选取，IDP k-means 算法中简单地将所有数据点均匀划分为 K 部分并求其中心的方法对聚类的优化并不够好，OEDP k-means 算法则是将所有数据点先按最近邻密度排序然后均匀划分为 K 部分并求其中心，该方法虽优于 IDP k-means 但当 K 个数据簇在高维空间对称分布且簇内密度均匀时得到的初始中心会趋于集中而发生劣化，而本文算法充分利用了计算离群点判定阈值时生成的距离矩阵，选取的初始聚类中心可以更接近实际的中心点，从而减少了聚类过程的迭代次数，也避免了过多噪声的引入，因此在实验中较 OEDP k-means 和 IDP k-means 算法表现更好。因此，依靠更合理的离群点处理和初始聚类中心选取策略，对于同等级的差分隐私保护，本文算法在聚类结果的可用性以及聚类效率等方面相比其他三种算法有更优的表现。

其次，考察在不同数据量下多节点并行处理对本文算法的加速情况。在 HTRU2 数据集上截取不同数量的记录集作为待处理数据集。对每一次截取所得数据集，单机运行本文算法，单机运行 OEDP k-means 算法，3 个子节点运行本文算法各 5 次取运行时间的平均值。这里的运行时间包含算法中的聚类初始化过程（离群点消除与初始聚类中心选取）耗时。具体图 5 所示。

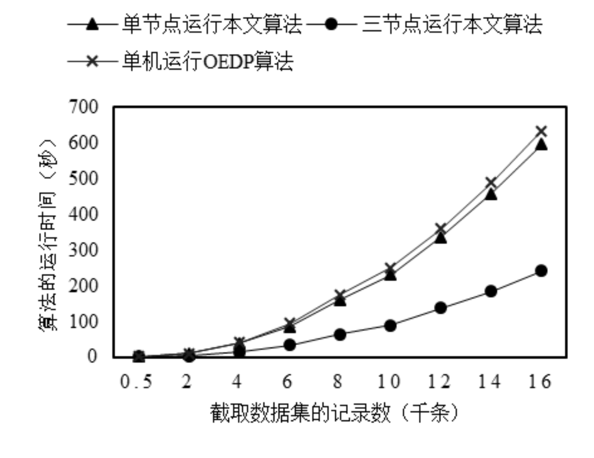


图 5　HTRU2 数据集上各情况的执行效率

由图 5 可知，单机运行下本文算法的耗时较 OEDP k-means 算法更低，这主要是因为本文算法对离群点的判定更为简洁，且对初始聚类中心的选取策略更为合理。而 3 节点下的本文算法耗时相比单机时有显著下降，这说明 MapReduce 框架下的本文算法可以很好地通过多节点并行计算来提高聚类算法的执行效率。另外，图中三种情况的耗时基本和截取数据集中的记录数呈二次曲线，这主要是由于为完成离群点的判定，需要计算数据集中各记录点的欧氏距离矩阵，而该过程的计算复杂度下界关于数据集中的记录数是二次的。虽然在获得图 5 数据的过程中每一次运行都会计算一遍距离矩阵，但在实际中，同一数据集上的初始化过程计算一遍即可。

4　结束语

本文基于 MapReduce 计算框架提出了一种并行化的支持差分隐私保护和离群点消除的 K-means 聚类算法，给出了由数据集各点间的欧氏距离矩阵与最近邻超球半径导出离群点判定阈值的并行计算方案，以及在此基础上设计的满足差分隐私保护的初始聚类中心选取策略。理论分析与实验结果表明算法在隐私保护的有效性，聚类结果的可用性以及执行效率和可扩展性等方面取得了很好的平衡，相比于同类算法有较优的表现。在后续的研究中，计划尝试更多不同的隐私保护预算分配策略并探索本文算法的更多扩展和应用。

参考文献：

[1] Sweeney L. k-anonymity: a model for protecting privacy [J]. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 2002, 10 (05): 557-570.

[2] 熊平，朱天清，王晓峰. 差分隐私保护及其应用 [J]. 计算机学报, 2014, 37 (1): 101-122.

[3] Machanavajjhala A, Kifer D, Gehrke J, *et al.* l-diversity: privacy beyond k-anonymity [J]. ACM Trans on Knowledge Discovery from Data, 2007, 1 (1): 3-3.

[4] Wong R C W, Li Jiuyong, Fu A W C, *et al.* (α, k) -anonymity: an enhanced k-anonymity model for privacy preserving data publishing [C]// Proc of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2006: 754-759.

[5] Xiao Xiaokui, Tao Yufei. M-invariance: towards privacy preserving re-publication of dynamic datasets [C]// Proc of ACM SIGMOD international conference on Management of data. New York: ACM Press, 2007: 689-700.

[6] Wong R C W, Fu A W C, Wang Ke, *et al.* Minimality attack in privacy preserving data publishing [C]// Proc of the 33rd International Conference on Very Large Data Bases. [S. l.] : VLDB Endowment, 2007: 543-554.

[7] Dwork C. Differential privacy [C]// Proc of the 33rd International Colloquium on Automata, Languages and Programming. 2006: 1-12.

[8] Blum A, Dwork C, Mcsherry F, *et al.* Practical privacy: the SuLQ framework [C]// Proc of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. New York: ACM Press, 2005: 128-138.

[9] Dwork C. A firm foundation for private data analysis [J]. Communications of the ACM, 2011, 54 (1): 86-95.

[10] 李洪成，吴晓平，陈燕. MapReduce 框架下支持差分隐私保护的 K-means 聚类方法 [J]. 通信学报, 2016, 37 (02): 124-130.

[11] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51 (1): 107-113.

[12] Yu Qingying, Luo Yonglong, Chen Chuanming, *et al.* Outlier-eliminated K-means clustering algorithm based on differential privacy preservation [J]. Applied Intelligence, 2016, 45 (4): 1179-1191.

[13] 李杨，郝志峰，温雯，等. 差分隐私保护 K-means 聚类方法研究 [J]. 计算机科学, 2013, 40 (3): 287-290.

[14] Dwork C. Differential privacy in new settings [C]// Procs of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, 2010: 174-183.

[15] Dwork C, Mcsherry F, Nissim K, *et al.* Calibrating noise to sensitivity in private data analysis [C]// Theory of Cryptography. Berlin: Springer, 2006: 265-284.

[16] Mcsherry F D. Privacy integrated queries: an extensible platform for privacy-preserving data analysis [C]// Proc of ACM SIGMOD International Conference on Management of data. New York: ACM Press, 2009: 19-30.

[17] Celebi M E, Kingravi H A, Vela P A. A comparative study of efficient initialization methods for the K-means clustering algorithm [J]. Expert Systems with Applications, 2013, 40 (1): 200-210.

[18] Aggarwal C C. Outlier analysis [C]// Data Mining. [S. l.] : Springer International Publishing, 2015: 237-263.

[19] Angiulli F, Fassetti F. Dolphin: an efficient algorithm for mining distance-based outliers in very large datasets [J]. ACM Trans on Knowledge Discovery from Data, 2009, 3 (1): 4.